REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE Technical Report	3. DATES COVERED (From - To) -

4. TITLE AND SUBTITLE Steganography Detection Using Entropy MeasuresTechnical Report	5a. CONTRACT NUMBER W911NF-11-1-0174
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 206022
6. AUTHORS Eduardo Melendez	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES Polytechnic University of Puerto Rico 377 Ponce de Leon Ave. Hato Rey, PR 00918 -0000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO 11. SPONSOR/MONITOR'S REPORT NUMBER(S) 58924-CS-REP.43

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
Steganography is the science of hiding the fact that some communication is taking place.
In general encryption, encoding and decoding are not required to accomplish steganography.
However, encryption serves as a layer of protection when steganography fails.
The first objective of steganography is hiding the existence of data exchange between two parties. In order to achieve this beyond any mere manipulation of the mean or

15. SUBJECT TERMS
Steganography, entropy measures, steganalysis, embedding techniques

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Alfredo Cruz
UU	UU	UU	UU		19b. TELEPHONE NUMBER 787-622-8000

Report Title

Steganography Detection Using Entropy MeasuresTechnical Report

ABSTRACT

Steganography is the science of hiding the fact that some communication is taking place. In general encryption, encoding and decoding are not required to accomplish steganography. However, encryption serves as a layer of protection when steganography fails.
The first objective of steganography is hiding the existence of data exchange between two parties. In order to achieve this beyond any mere manipulation of the mean or carrier, the existence of information exchange must be kept away from the reach of human radar sensors. Second, steganography must cause little or no impact on the carriers structure. The latter guarantees to prevent suspicious of some sort of manipulation. Third, carriers must outweigh message existence, i.e., it must be robust. The medium should withstand a certain level of modification before data existence is detected. Finally, the capacity of the medium should allows to handle a certain level of information before hidden information is detected.
Steganography is studied taking under consideration the capabilities of detection of the information transferred. This include steganalysis, the techniques and methods used to detect steganography. The importance of the performance of steganography will lead, either from simple to more complex methods of detection. The importance of embedding matches the efficiency of the detection technique.
There are two problems in steganalysis: (1) detecting the existence of a hidden message and (2) decoding the message. As terrorist groups have been known to use steganography in planning their attacks, this has become an important problem of national security. This technical report is only concerned first, with embedding techniques and second, the problem of hidden message detection using steganalysis.
The approach is to statistically analyse the least significant bit(s) of each color dimension of each pixel to look for some kind of a pattern. In the absence of a hidden message this should look like random noise. Addition of a hidden message will affect the entropy of the data. This difference should be detectable by comparing the entropy of unaltered picture files with the entropy of files with embedded steganography. Many software are available in the market for steganography and steganalysis. However, in order to better understand steganography we have developed simple procedures for embedding. The programming tools used for this purpose are mainly the R-Language because of the broad scope and strength in the statistical analysis. A second programming language used was C# for its well structured user interface developer tools.

Steganography Detection Using Entropy Measures

Technical Report

By Eduardo Meléndez

Universidad Politécnica de Puerto Rico

November 16, 2012

DoD HBCU # W911NF-11-1-0174:

"Enhancing Research in Networking & System Security, and Forensics, in Puerto Rico"

Alfredo Crúz, PhD
Jeff Duffany, PhD
Eduardo Meléndez, MS

This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number: W911NF1110174.

melendez.eduardo@gmail.com

Contents

Part I. Introduction

1. Introduction

Steganography is the science of hiding the fact that some communication is taking place. In general encryption, encoding and decoding are not required to accomplish steganography. However, encryption serves as a layer of protection when steganography fails.

The first objective of steganography is hiding the existence of data exchange between two parties. In order to achieve this beyond any mere manipulation of the mean or carrier, the existence of information exchange must be kept away from the reach of human radar sensors. Second, steganography must cause little or no impact on the carriers structure. The latter guarantees to prevent suspicious of some sort of manipulation. Third, carriers must outweigh message existence, i.e., it must be robust. The medium should withstand a certain level of modification before data existence is detected. Finally, the capacity of the medium should allows to handle a certain level of information before hidden information is detected.

Steganography is studied taking under consideration the capabilities of detection of the information transferred. This include steganalysis, the techniques and methods used to detect steganography. The importance of the performance of steganography will lead, either from simple to more complex methods of detection. The importance of embedding matches the efficiency of the detection technique.

There are two problems in steganalysis: (1) detecting the existence of a hidden message and (2) decoding the message. As terrorist groups have been known to use steganography in planning their attacks, this has become an important problem of national security. This technical report is only concerned first, with embedding techniques and second, the problem of hidden message detection using steganalysis.

The approach is to statistically analyse the least significant bit(s) of each color dimension of each pixel to look for some kind of a pattern. In the absence of a hidden message this should look like random noise. Addition of a hidden message will affect the entropy of the data. This difference should be detectable by comparing the entropy of unaltered picture files with the entropy of files with embedded steganography. Many software are available in the market for steganography and steganalysis. However, in order to better understand steganography we have developed simple procedures for embedding. The programming tools used for this purpose are mainly the R-Language because of the broad scope and strength in the statistical analysis. A second programming language used was C# for its well structured user interface developer tools.

We have outlined the steps to proceed with the development of our research. These are follows:

- Obtain sample jpegs from the Internet or other source
- Import these sample files as data files into R Language
- Statistically analyse least significant bits.
- Use steganography to hide messages in a sample of jpeg files.
- Import as a data file into R language and statistically analyse the least significant bits of the jpeg files with known hidden messages.
- Compare with original file in terms of entropy.

2. History

There are many instances with particular purposes where steganography has been used. In particular there are three instances that are mentioned generally mentioned.

From the ancient times, the Greek historian Herodotus writes of a nobleman, Histaeus, who needed to communicate with his son-in-law in Greece. He shaved the head of one of his most trusted slaves and tattooed the message onto the slave's scalp. When the slave's hair grew back the slave was dispatched with the hidden message.

During the Second World War the Microdot technique was developed by the Germans. Information, especially photographs, was reduced in size until it was the size of a typed period. Extremely difficult to detect, a normal cover message was sent over an insecure channel with one of the periods on the paper containing hidden information.

Today steganography is mostly used on computers with digital data being the carriers and networks being the high speed delivery channels.

Definitions are important to understand concepts and we may say that steganography is the art of hiding the fact that communication is taking place, by hiding information in other information or mean.

2.1. Steganography concepts

Although steganography is an ancient subject the modern formulation is being focus in two ideas, passive and active. An example of a warden who has knowledge of the communication between two inmates. This view was proposed by Simmons, where two inmates wish to communicate in secret to hatch a plan to escape from prison. Their communication passes through the warden who will throw them in solitary confinement should she suspect any convert communication.

The warden, who is free to examine all communication exchanged between the inmates, can either be passive or active.

- passive: A passive warden simply examines the communication to try and determine if it potentially contains secret information. If she suspects a some communication to contain hidden information, a passive warden takes note of the detected covert communication, reports this to some outside party and lets the message through without blocking it.
- active: An active warden, on the other hand,, will try to alter the communication with the suspected hidden information deliberately, in order to remove the information.

Knowing the existence of some communication between two parties allows you to either, alter the information contained and passed through the medium or just let it pass through.

2.2. Steganography Vs Cryptography

There are many differences between steganography and cryptography. None of them need of each other to execute its purpose. However, both do coexist and furthermore, they become a powerful security tool when used together.

Steganography differs from cryptography in the sense that where cryptography focuses on keeping the contents of a message secret, stagenography focuses on keeping the existence of a message secret.

Steganography and cryptography are both ways to protect information from unwanted parties but neither technology alone is perfect and can be compromised.

Once the presence of hidden information is revealed or even suspected, the purpose of steganography is partly defeated.

The strength of steganography can thus be amplified by combining it with cryptography.

Two other technologies that are closely related to steganography are watermarking and fingerprinting. These technologies are mainly concerned with the protection of intellectual property, thus the algorithms have different requirements than steganography.

2.3. Different kinds of steganography

There are four types of categories of steganography. Almost all digital file formats can be used for steganography, but the formats that are more suitable are those with a high degree of redundancy. Redundancy can be defined as the bits of an object that provide accuracy far greater than necessary for the object's use and display. The redundant bits of an object are those bits that can be altered without the alteration being detected easily. Image and audio files especially comply with this requirement, while research has also uncovered other file formats that can be used for information hiding.

- Text: An obvious method was to hide a secret message in evry n^{th} letter of every word of a text message.

- Images: Given the large amount of redundant bits present in the digital representation of an image, images are the most popular cover objects for steganography.
- Audio / Video: A different technique unique to audio steganography is masking, which exploits the properties of the human ear to hide inforamtion unnoticeable. A faint, but audible, sound becomes inaudible in the presence of another louder audible sound. This property creates a channel in which to hide information.
- Protocol: The term protocol steganography refers to the technique of embedding information within messages and network control protocols used in network transmission. In the layers of the OSI network model there exist convert channels where steganography can be used.

Part II. Steganography

3. Images and Significance of Bits

The main object of Steganography is the fact that communication is being occurring with out attracting attention. Information is being exchanged hidden, but from the interested parties, i.e. the sender and the receiver. However, once the communication is been compromised, steganography simply fails.

Information is exchanged between two or more parties through a communication medium. These ranges from text, images, videos and more complex ones. In our research the mean of communication to be used is related to images, in particular we are going to use jpeg types. The reason for this is that are the most common and currently used image format. Furthermore, its configuration is simple to manipulate.

3.1. Image definition

Images are very useful means to hide messages (embedding). The mechanism of embedding is accomplished by manipulating certain bits in the binary color representation. A monochrome picture is depicted in different scales of gray including black and white. Each pixel is a byte (a string of 8 bits). From black to white we have $2^8 = 256$ different tones of gray. These range from white, 00000000, through black, 11111111. A given message with proper size can be embedded in a cover (image) by manipulating the bits on each pixel. Let us assume that in a particular pixel the gray is represented by 00001111. By switching the second bit we obtain a new binary string, 01001111. The latter change has modified the original picture.

Colour pictures or RGB images are set in each pixel with a binary string representation of length 24. The first 8 bits represent the red color, green by the next 8 bits and blue by the last 8 bits. Each pixel ranges in colors from white to black, including combinations of red, green and blue shades. In total, there are $256^3 = 16,777,216$ possibilities of color shades. We show below three sets of 8 bits-string. Each set represents a color. From an original image, say

$$00001010 \quad 00110101 \quad 00011110$$

we can change the 4^{th} bit from left to right, obtaining

$$00011010 \quad 00100101 \quad 00001110$$

The latter by no means is equivalent to steganography. As we will see later on there is a big difference between changing the *least significant bit* (lsb) and changing the most significant bit.

3.2. Image Compression

When working with large images, we start having problems handling large files. Some sort of compression is necessary in order to better handle these images. There are two types of compression: lossy and lossless. An example of the first type of compression technique is JPEG (Joint Photographic Experts Group) image format. For the second type, we have the GIF (Graphical Interchange Format) and the 8-bit BMP (Microsoft Windows Bitmap file). In the first case loss of information takes place, while in the second the integrity of the original information remains.

The process of jepg compression requires the calculation of discrete cosine transforms coefficients and a quantization matrix. The latter leads to the level of compression of the image.

3.3. Least Significant Bit (LSB)

The object of steganography is to prevent suspicion upon the action of communication, regardless the mean being used. Small changes in the tone of gray will in general, be imperceptible to the human eye. The Least Significant Bit (LSB) is a simple approach to modify an image, while at the same time making the change imperceptible to the human eye. By considering the redundant bits (least significant bits), imperceptible changes take place by changing the 8^{th} (from left to right) bit in the string of eight bits. For example, by changing 00001111 to 00001110, we have applied the least significant bit technique.

3.4. Significant Bit Image Depiction

Steganography fails to comply in its purpose at the very moment when the existence of the message has been compromised. Even when steganography is not infallible, its strength lies entirely on the non-knowledgeable of its content, whatever it is. When the mean of communication is a picture from which a text or a message can be extracted, its infallibility is directly related to the manipulation of the pixels. In particular, by manipulating the LSB, any message is safe as long as it remains imperceptible to the human eye. The following pictures show the level of *visual perception* in relation to the change of bits in each pixel for each channel (color) in each pixel. An image is compared before and after the LSB technique has been applied. The simple procedure of switching all LSB for each pixel in each channel produces a different image that cannot be distinguished from the original one. Figure 1 depicts side by side two pictures. The original picture from the left has been altered using the LSB technique resulting in the picture from the right. We are unable to perceive any changes in the image after the LSB technique.

Figure 2 shows two images. From the left is the resultant image after switching bit number 2 in each pixel for each channel. The image from the right is obtained by switching bit number 3. In the latter image a gradual change in color is being noticed.

Changes made from the bits number 4 and 5 are shown in Figure 3, left and right, respectively. The manipulation is evident in both pictures. Note that colors are distorted and degraded.

Finally, the switch of bit number 8 in all channels for every pixel makes the modification evident. So it is that it can be perceived the by human eye. These bits (number 8) are extremely significant, and if steganography is the intended purpose, would be unwise to choose bit number 8. Below, we show the original image side by side with the 8^{th}-bit-switch depiction.

The importance in using the LSB is addressed to preserve the objective of steganography, i.e., to hide the fact that communication is taking place.

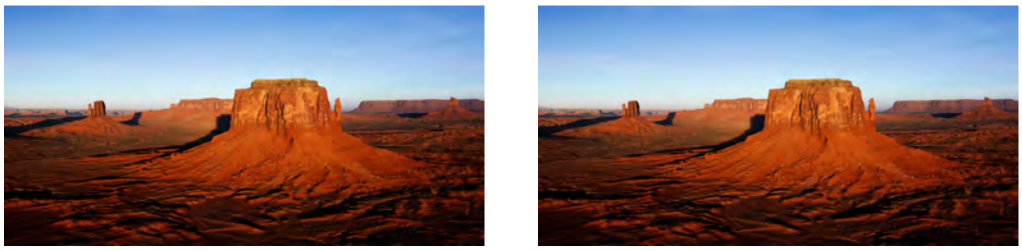


Figure 1: Original Depiction (left); LSB Depiction (right)

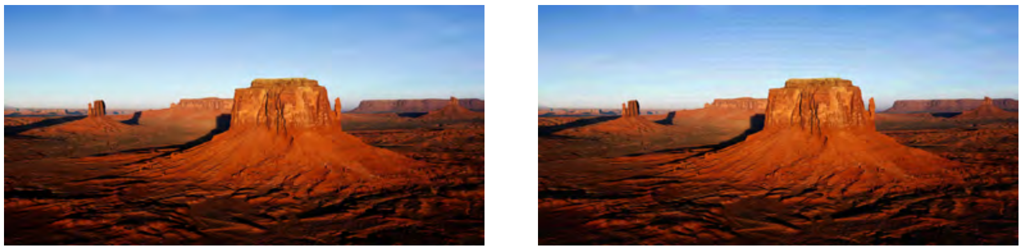


Figure 2: Switch of bit number 2 (left); Switch of bit number 3 (right)

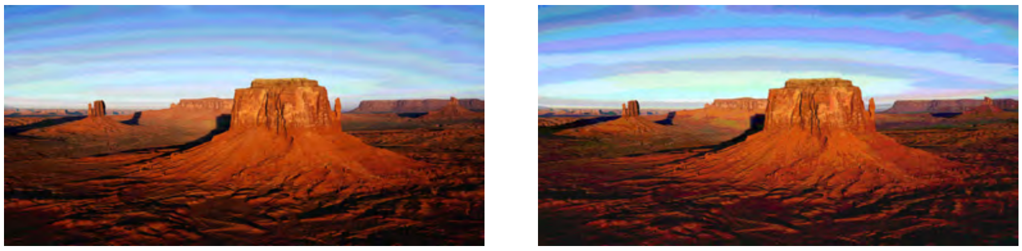


Figure 3: Switch of bit number 4 (left); Switch of bit number 5 (right)

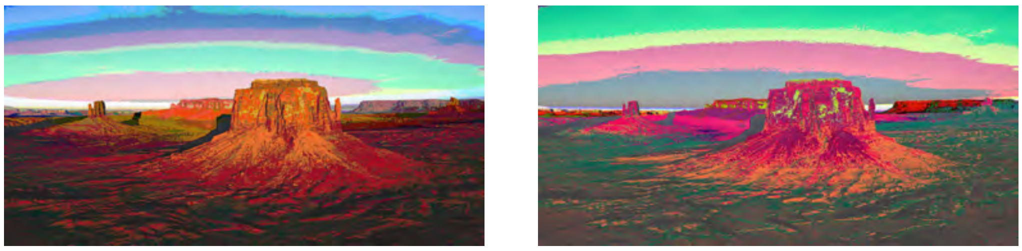


Figure 4: Switch of bit number 6 (left); Switch of bit number 7 (right)

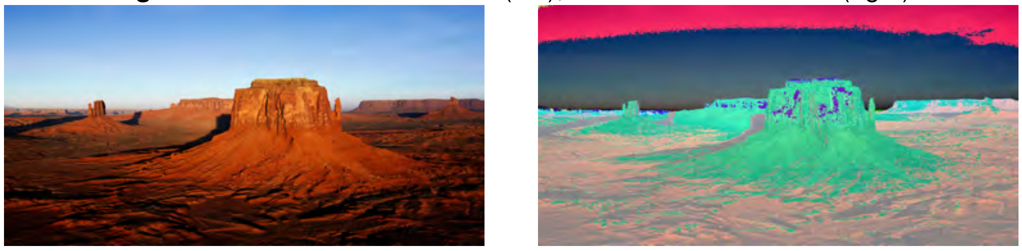


Figure 5: Original depiction (left); Switch of bit number 8 (right)

4. Steganography Ad Hoc Methods

There are probably a finite number of ways of embedding a message in an image. The embedding is possible according to the length of the message and the dimensions of the image. Because secrecy is the objective of Steganography, the act of communicating must be kept hidden. Below we will discuss some embedding methods. These are not the most efficient, but are used to accomplish one of the task of Steganography, embedding a message.

We may consider various strategies for embedding. The latter must be imperceptible to human eye. The technique must take place sequentially, at least for now. In other words, there is no randomness (apparent) in the way a message is embedded, there exist an order. One general method is to embed a message line by line. For this, we may discuss many approaches. We are going to limit ourselves to three methods, line-by-line, uniformly distributed and analogously to the LSB method, LSBP, or the less significant bits per pixel.

4.1. Linear Ordered Embedding

4.1.1. Line-by-Line Linear Embedding

Probably the most obvious and simple method of embedding is line-by-line. With appropriate message length and image dimensions, this strategy will be achieved. The length of the message must be within the total number of bytes (8 bits) contained by the image along the three channels. In other words, if l is the length of the message and $m \times n \times 3$ the dimensions of the image, $l \leq 3mn$ must be satisfied.

The general technique is very simple. The message is written by channels, from left to right and from top to bottom. By selecting a channel, say k, either you may start from the most top-left position of that color-matrix, $(1, 1, k)$, or from some point (i, j, k) in that matrix. Continue embedding the message from that point on, from left to right and from top to bottom.

The graph below shows the trace of the embedded message when this is written.

1,~~1~~	~~1,2~~	~~1,3~~	~~1,4~~	~~1~~,5
2,~~1~~	~~2,2~~	~~2,3~~	~~2,4~~	~~2~~,5
3,~~1~~	~~3,2~~	~~3~~,3	3,4	3,5
4,1	4,2	4,3	4,4	4,5
5,1	5,2	5,3	5,4	5,5

he algorithm starts with the embedding in a channel. Remember that a channel is precisely the matrix of color's shade. Let us assume that we are starting at the top-left most pixel. In a monochrome picture we are dealing with different shades of gray, including white and black. We start determining if the length of the message do not exceed the dimensions of the matrix. If the length of the message is feasible, in the sense that the length fits in the matrix dimensions, then we substitute each pixel represented by 8 bits, by the binary representation of each character in the message. We continue doing these substitution linearly, until the end of the message. These substitutions take place one pixel after another, skipping none.

Algorithm

Let M a matrix of dimension $r \times c$. Let $s = \{s_1, s_2, ..., s_l\}$ the message to be embedded of length $|s| = l$, i.e. the number of characters in the sentence. Let $l \equiv k \bmod c$ with $0 \leq k < c$. Let $f = \frac{l-k}{r} + 1$, the minimum number of rows needed to write the message. In the most simple case, consider an image, a monochrome type of dimension $r \times c$. Let partition our message $s = \{s_1^*, s_2^*, ..., s_{f-1}^*, s_f^*\}$ in segments of equal length $|s_i^*| = c$, for $i = 1, 2, ..., f-1$, with the exception of the last one which is bounded, $|s_f^*| \leq c$.

Given the matrix

$$M = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_r \end{bmatrix}.$$

The rows $\{t_1, ..., t_{f-1}\}$ are substituted by $\{s_1^*, s_2^*, ..., s_{f-1}^*\}$. The last segment is embedded in row t_f, so that the first $|s_f^*|$ elements of t_f are those of s_f^*. Let us label the latter row by t_f^*. Our covert matrix is then given by

$$M_C = \begin{bmatrix} s_1^* \\ s_2^* \\ \vdots \\ t_f^* \\ t_{f+1} \\ \vdots \\ t_r \end{bmatrix}.$$

An algorithm is built so that from the matrix M, one obtain the matrix M_C. The algorithm, even when is simple, it is inefficient. The embedding is accomplish by the substitution of a character code (ascii or binary representation) into the binary of the pixel. The following algorithm has been built for monochrome images embedding.

```
function($M, s$)
{
    for(i in 1:$f$)
    {
        if( $i < f$)
        {
            $t_i = s_i^*$
        }
        else
        {
            $t_{(f,|s_f^*|)} = s_f^*$
        }
    }
    return $M$
}
```

4.1.2. Uniform Spread Embedding (USE)

The uniform spread of a message along the matrix M is another form of linear embedding. The message is spread over the matrix in a orderly fashion. Given a message s and a matrix M, with dimension $r \times c$ the length of the message $|s| \leq rc$. Let us take the floor $k = \lfloor rc/|s| \rfloor$. Starting from the most top-left pixel, if one character s_m is to be substituted at pixel (i, j), $i, j = 1, 2, \ldots, n$, the next character will be embedded at $(\phi_r(i, j, k), \phi_c(i, j, k))$. The functions ϕ_r and ϕ_c return the corresponding indices of the pixel from which embedding is going take place.

Algorithm

In R we can implement the algorithm as follow. Consider the array $a = a_1a_2...a_r$, obtained from the rows of the matrix M, by putting the rows a_m sequentially from top to bottom and from left to right. Note that $|a| = rc$. Starting from 1, we are going to substitute sequentially a character of the message s in a every k steps. A new matrix

N is built with the resulting substitutions. Using the R-language we may accomplished our goal as follows.

```
function($M, s$)
{
    $a$ = c(t($M$))
    $k$ = floor($|a|/|s|$)
    $a$[seq(1,$|a|, k$)] = $s$
    $N$ = matrix($a$,nrow=$r$,ncol=$c$,byrow=TRUE)
    return $N$
}
```

The functions ϕ_r and ϕ_c provide the indices for the next embedding. The function ϕ_r is outlined as follows.

```
function($i, j, k$)
{
    if($j + k <= c$)
    {
      return $i$
    }
    else
    {
      return $i + 1$
    }
}
```

For the function ϕ_c we have

```
function($i, j, k$)
{
    if($j + k <= c$)
    {
      return $j + k$
    }
    else
    {
      return $(j + k - 1)\%c + 1$
    }
}
```

4.1.3. Pixel LSBP

Another technique for embedding is the LSBP (less significant bits per pixel). This take each character in a message s, and spread it over the less significant bits of a pixel. In the case of a monochrome image, this technique becomes the line-by-line linear technique. However, in a RGB image the binary representation of a character is split in combinations of 3,3,2 or 3,2,3 or 2,3,3 bits, to be distributed in channels Red, Green and Blue, respectively.

Consider the character s_i, from the message s. Its binary representation $(x_1, x_2, ..., x_8)$ will be embedded in the pixel (v, w). This pixel has three bytes

$$00101011 \quad 01101111 \quad 11001000$$

containing each color shade. By substituting our character and using configuration 3,3,2, we obtain

$$00101x_1x_2x_3 \quad 01101x_4x_5x_6 \quad 110010x_7x_8$$

Algorithm

Using R coding we make the embedding per pixel. In the algorithm below the pixel p and a message s (in a binary representation) are sent for the embedding.

```
function($p, s$)
{
    $pr < -toBin(p[v, w, 1])$
    $pg < -toBin(p[v, w, 2])$
    $pb < -toBin(p[v, w, 3])$
    $pr[6 : 8] < -s[1 : 3]$
    $pg[6 : 8] < -s[4 : 6]$
    $pb[7 : 8] < -s[7 : 8]$
    $p[v, w, 1] < -toInt(pr)$
    $p[v, w, 2] < -toInt(pg)$
    $p[v, w, 3] < -toInt(pb)$
    return $p$
}
```

The latter algorithm is not complete, of course. This is part of a sequel of steps. This could be used with a modification of either the line-by-line or the USE techniques.

4.1.4. LSB

All the techniques shown so far suffer of a great flaw, they make steganography to fail. With a wide and large monitor, 32-in maybe or larger, a line of dots may be seen across the lines. These dots are the change of color shades caused by the embedding. The change is visible and evident, violating the essence of the purpose of steganography.

In order to prevent this flaw, we are going to use the LSB technique. In the case of the monochrome image, the binary representation of a character is distributed along 8 pixels. The following character is spread over the next 8 pixels, and so on. The weakness of this technique is that the length of the message cannot be considerable large. In this case we have space constraints. The message is forced not to exceed $|s|/3 \leq 3rc$, or better put $|s| \leq 3rc$ in a RGB picture, and $|s| \leq rc$ in a monochrome image.

Algorithm

We can combine a number of techniques that lead us to a considerable level of efficiency.

```
function($M, s$)
{
    $N = c(toBin(c(t(M))))$
    $m = c(toBin(s))$
    #$k$ is a stepping constant
    $N$[seq(8,$length(N), k$)] = $m$
    $N$ = matrix($N$,nrow=$r$,ncol=$c$,byrow=TRUE)
    return $N$
}
```

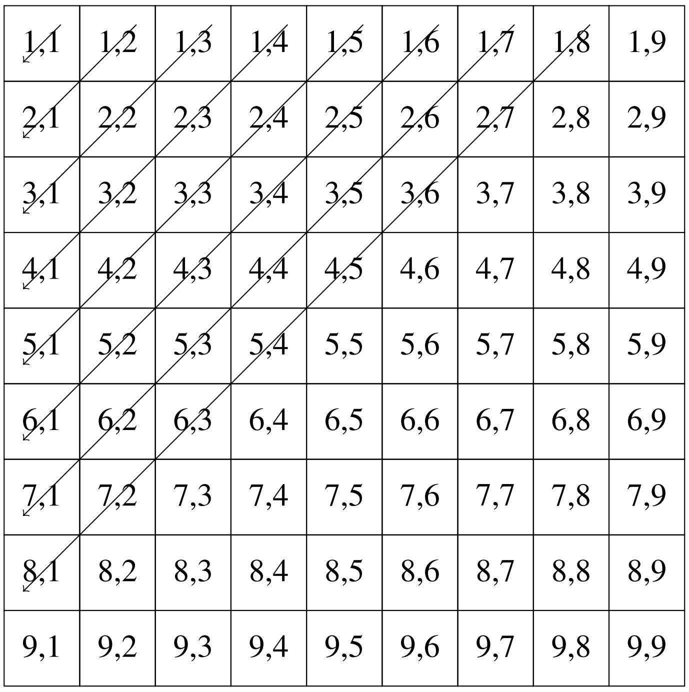
The above algorithm shows the technique of spreading and the use of the LSB.

4.2. Diagonal

Different from linear techniques, our embedding can be traced diagonally. We are going to mention four main techniques and its variants, Top-To-Bottom-Right-To-Left, Top-To-Bottom-Left-To-Right, Bottom-To-Top-Right-To-Left, Bottom-To-Top-Left-To-Right. In total there are different cases.

4.2.1. Top-To-Bottom-Right-To-Left-Left Corner

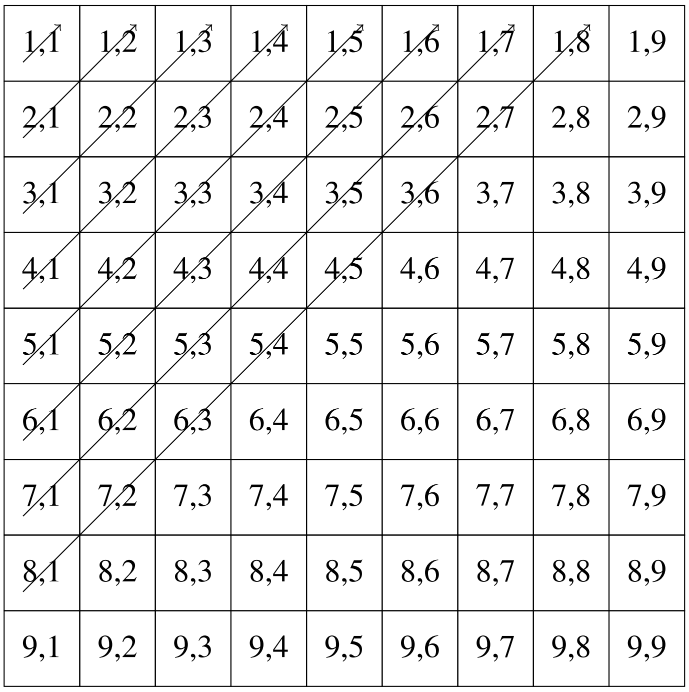
We can start our embedding according to the following sequence of indices {(1,1), (1,2), (2,1), (1,3), (2,2), (3,1),...}. The following diagram shows more details.



1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9
5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9
6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9
7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9
8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8	8,9
9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8	9,9

4.2.2. Top-To-Bottom-Left-To-Right-Left Corner

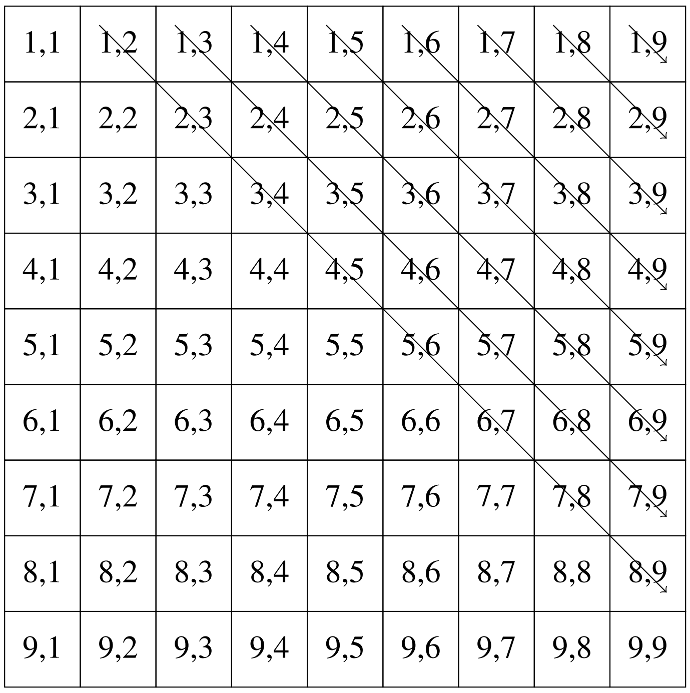
We can start our embedding according to the following sequence of indices {(1,1), (2,1), (1,2), (3,1), (2,2), (1,3),...}. The following diagram shows more details.



1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9
5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9
6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9
7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9
8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8	8,9
9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8	9,9

4.2.3. Top-To-Bottom-Left-To-Right-Right Corner

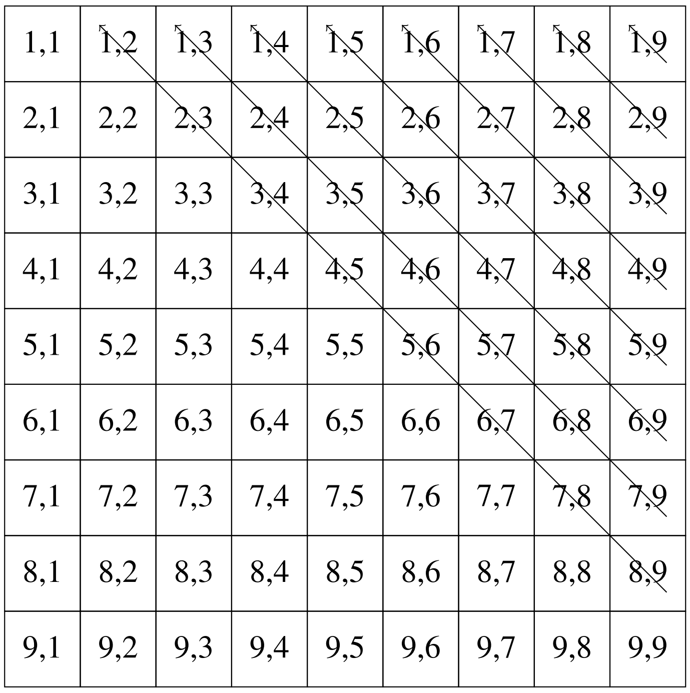
We can start our embedding according to the following sequence of indices {(1,c), (1,c-1), (2,c), (1,c-2), (2,c-1), (3,c),...}. The following diagram shows more details.



1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9
5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9
6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9
7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9
8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8	8,9
9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8	9,9

4.2.4. Top-To-Bottom-Left-To-Right-Right Corner

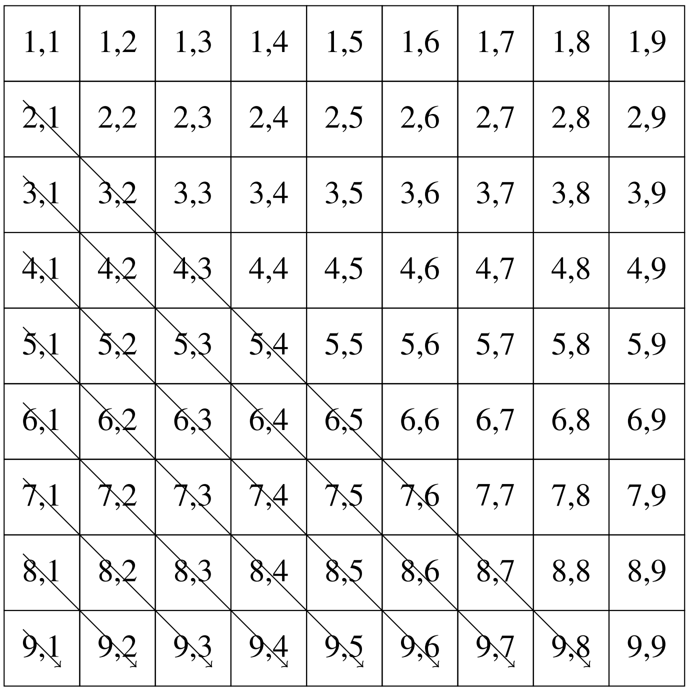
We can start our embedding according to the following sequence of indices {(1,c), (2,c), (1,c-1), (3,c), (2,c-1), (1,c-2),...}. The following diagram shows more details.



1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9
5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9
6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9
7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9
8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8	8,9
9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8	9,9

4.2.5. Bottom-To-Top-Left-To-Right-Left Corner

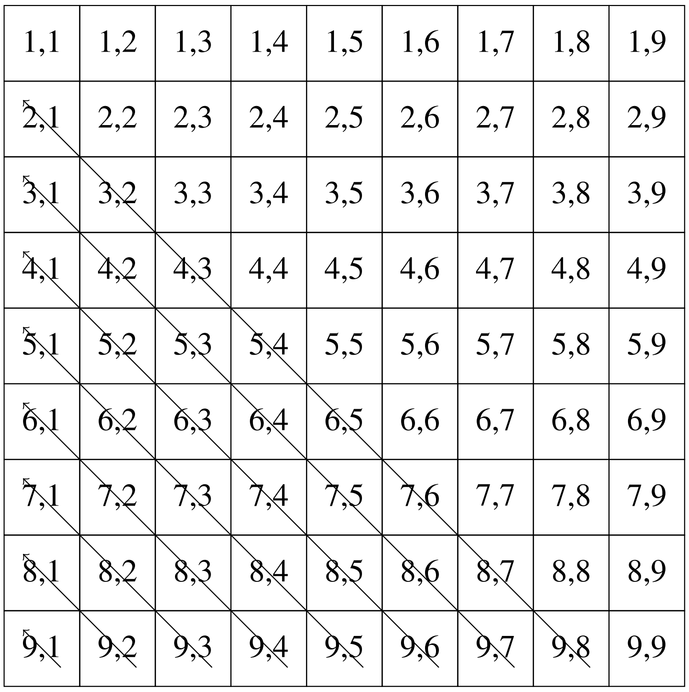
We can start our embedding according to the following sequence of indices {(r,1), (r-1,1), (r,2), (r-2,1), (r-1,2), (r,3),...}. The following diagram shows more details.



1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9
5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9
6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9
7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9
8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8	8,9
9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8	9,9

4.2.6. Bottom-To-Top-Right-To-Left-Left Corner

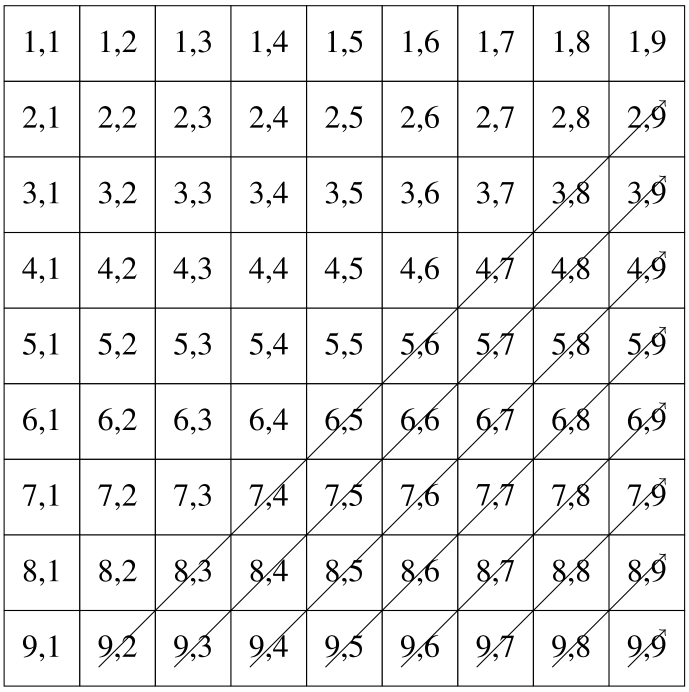
We can start our embedding according to the following sequence of indices {(r,1), (r,3), (r,2), (r-1,1), (r-1,2), (r-2,1),...}. The following diagram shows more details.



1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9
5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9
6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9
7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9
8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8	8,9
9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8	9,9

4.2.7. Bottom-To-Top-Left-To-Right-Right Corner

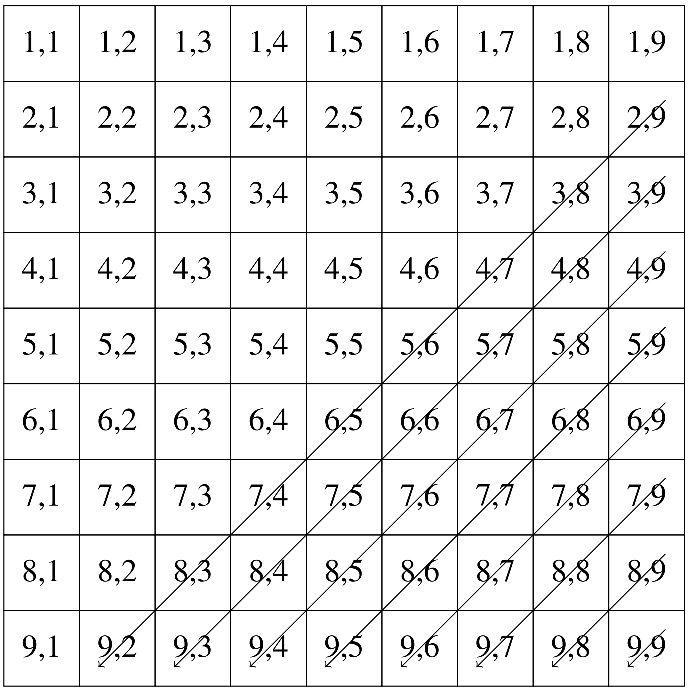
We can start our embedding according to the following sequence of indices {(r,c), (r,c-1), (r-1,c), (r,c-2), (r-1,c-1), (r-2,c),...}. The following diagram shows more details.



1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9
5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9
6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9
7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9
8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8	8,9
9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8	9,9

4.2.8. Bottom-To-Top-Right-To-Left-Right Corner

We can start our embedding according to the following sequence of indices {(r,c), (r-1,c), (r,c-1), (r-2,c), (r-1,c-1), (r,c-2),...}. The following diagram shows more details.



1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9
5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9
6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9
7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9
8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8	8,9
9,1	9,2	9,3	9,4	9,5	9,6	9,7	9,8	9,9

Part III. Steganalysis

5. Discrete Cosine Transform (DCT)

5.1. DCT coefficients in general

The Discrete Cosine Transform (DCT) is used to transform values from successive 8×8 pixels from the image (this being set to appropiate values) to a block of DCT coefficients of same dimensions. The DCT coefficients can be obtained using a Type II DCT table given by the equation below for an integer N

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} m(x,y) d_N(x,i) d_N(y,j) \tag{1}$$

where

$$d_N(a,b) = \cos\left[\frac{(2a+1)b\pi}{2N}\right] \quad \text{and} \quad C(n) = \begin{Bmatrix} \frac{1}{\sqrt{2N}} & if & n = 0 \\ 1 & if & n > 0. \end{Bmatrix} \tag{2}$$

and $m(x,y)$ are the entries from a matrix m. Type II is the most frequently used in many applications.

5.2. DCTs of a 8×8 block matrix

In the particular case when $N = 8$, for a block 8×8, the equation is reduced to

$$D(i,j) = \frac{1}{4} C(i)C(j) \sum_{x=0}^{7} \sum_{y=0}^{7} m(x,y) d_N(x,i) d_N(y,j) \tag{3}$$

with

$$d_N(a,b) = \cos\left[\frac{(2a+1)b\pi}{16}\right] \quad \text{and} \quad C(n) = \begin{Bmatrix} \frac{1}{4} & if & n = 0 \\ 1 & if & n > 0. \end{Bmatrix} \tag{4}$$

The matrix T obtained from equation (3) is shown below.

```
           [,1]        [,2]        [,3]        [,4]        [,5]        [,6]        [,7]        [,8]
[1,] 0.35355339  0.35355339  0.35355339  0.35355339  0.35355339  0.35355339  0.35355339   0.35355339
[2,] 0.49039264  0.41573481  0.27778512  0.09754516 -0.09754516 -0.27778512 -0.41573481  -0.49039264
[3,] 0.46193977  0.19134172 -0.19134172 -0.46193977 -0.46193977 -0.19134172  0.19134172   0.46193977
[4,] 0.41573481 -0.09754516 -0.49039264 -0.27778512  0.27778512  0.49039264  0.09754516  -0.41573481
[5,] 0.35355339 -0.35355339 -0.35355339  0.35355339  0.35355339 -0.35355339 -0.35355339   0.35355339
[6,] 0.27778512 -0.49039264  0.09754516  0.41573481 -0.41573481 -0.09754516  0.49039264  -0.27778512
[7,] 0.19134172 -0.46193977  0.46193977 -0.19134172 -0.19134172  0.46193977 -0.46193977   0.19134172
[8,] 0.09754516 -0.27778512  0.41573481 -0.49039264  0.49039264 -0.41573481  0.27778512  -0.09754516
```

This matrix is an orthogonal matrix, i.e., $T^{-1} = T'$ or $TT' = T'T = I$, where I is the identity matrix.

The implementation of the equation above for a given image (gray scale), M, by taking a 8$\times$ 8 block M_{xy} the DCT coefficients matrix (for a block) is given by

$$D_{xy} = TM_{xy}T'$$

A block of a given dimension (8×8) contains values from ranging from 0 to 255. Since DCT domain runs over symmetric values above the origin, we level off our block by 128. The latter provides the symmetry required upon the images' values.

These values are called the un-quantized DCT coefficients and our matrix is in the DCT domain.

5.3. DCT coefficients of an image

In general we can obtain DCT coefficients for any clock of any dimension. One aspect to achieve the DCT coefficients matrix over the whole image is by partitioning the image in equal dimensional blocks. The latter task results in many loops and time consuming. We can construct block diagonal matrices that will lead us to the DCT coefficients image. Two matrices will be used, one for the left T_l and one for the right T_r multiplications. The latter are given by

$$T_l = \begin{bmatrix} T & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & T & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & T & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & T \end{bmatrix}_l \quad \text{and} \quad T_r = \begin{bmatrix} T & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & T & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & T & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & T \end{bmatrix}_r , \tag{5}$$

where

$$l = \frac{\text{\# of rows}}{8} \quad \text{and} \quad r = \frac{\text{\# of columns}}{8}.$$

T_l and T_r are block diagonal matrices which diagonal region are filled with the matrix T. There are precisely l and r of such block matrix T, in T_l and T_r, respectively.

Let us assume that the image M has appropriate values, i.e., level off from [0,255], with dimensions multiple of 8. The DCT coefficients matrix of the image

$$D = T_l M T_r', \tag{6}$$

results in consecutive DCT coefficients blocks of dimension 8$\times$8. The matrix D and M are of the form

$$D = \begin{bmatrix} D_{11} & D_{12} & \dots & D_{1r} \\ D_{21} & D_{22} & \dots & D_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ D_{l1} & D_{l2} & \dots & D_{lr} \end{bmatrix} \quad \text{and} \quad M = \begin{bmatrix} M_{11} & M_{12} & \dots & M_{1r} \\ M_{21} & M_{22} & \dots & M_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ M_{l1} & M_{l2} & \dots & M_{lr} \end{bmatrix}. \tag{7}$$

where $D_{xy} = TM_{xy}T'$, $D = (D_{xy})_{xy}$ and $M = (M_{xy})_{xy}$.

5.4. Quantization Index Modulation (QIM)

Basic principles

Consider the case of embedding a message m in a host signal $\mathbf{x} \in \Re^N$. This host signal can be a vector of pixel values of Discrete Cosine Transform coefficients from an image. We wish to embed at a rate of R_m bits per dimension (bits per host signal sample) so we can think of m as an integer, where

$$m \in \{1, 2, \ldots, 2^{NR_m}\}.$$

An embedding function $s(x, m)$, maps the host signal x and the information to embed m, to a composite signal $s \in \Re^N$. The embedding should not degrade the host signal, so that we have a minimum level or some distortion measure $D(s, x)$ between the composite and host signals.

There are two main or general classes to distinguish embedding methods from. First we have the host-interference nonrejecting methods and second, the host-interference rejecting methods.

Host-interference nonrejecting methods have the general property that the host signal is effectively a source of interference in a system. The simplest of such methods have purely additive embedding functions of the form

$$s(x, m) = x + w(m)$$

where $w(m)$ is typically a pseudo-noise sequence, often called as additive spread-spectrum method. It is very common to express $w(m) = a(m)v$, where v is a unit-energy spreading vector and $a(m)$ is a scalar function of the message.

5.5. Properties of an ensemble function

The embedding should not degrade the host signal up to a point that the embedding can be capture. Therefore, a distortion measure $D(s, x)$ between the composite and host signals should be relatively small. One might measure, in theory the difference between the host signal and the composite.

For example, one might consider the square-error distortion measure

$$D(s, x) = \frac{1}{N}||s - x||^2. \tag{8}$$

Even more, we can express the composite function as the sum of the host signal and nuisance, i.e.,

$$s(x, m) = x + e(x, m). \tag{9}$$

Nuisance might be expressed as

$$e(x,m) \triangleq s(x,m) - x \tag{10}$$

Now, to develop the QIM concept the mapping $s(x,m)$ can be viewed as an ensemble of functions of x, indexed by m. If the strength of the embedding function is to cause small distortion, then we may state the identity-approximate property

$$s(x,m) \approx x, \quad \forall m.$$

The fact that the system needs to be robust to perturbations suggests that the points in the range of one function in the ensemble should be far away in some sense from the points in the range of any other function.

Now, in reality, we don't know much about the original information, but the mapping s, i.e., either few or no information regarding the source values x and m.

The ideal would be the function ranges to have no common ground or intersection. Otherwise, some values of s will lead to some undetermined m. Precisely, the non-intersection property leads to host-signal interference rejection.

The non-intersection property along with the approximate-identity property, which suggests that the ranges of each of the functions "cover" the space of possible (or at least highly probable) host-signal values x, suggests that the functions be discontinuous.

Quantizers are just such a class of discontinuous, approximate-identity functions. QIM refers to embedding information by first modulating an index or sequence of indices with the embedded information and then quantizing the host signal with the associated quantizer or sequence of quantizers.

These quantizers are analysed from the probabilistic scope. For the following slides we will develop the probabilistic framework. The latter will allows us to define tests and non-parametric structures for stego image identification.

Let us define the probability framework. First, let χ the random space of possible values for the the host signal $\mathbf{x}$ or *cover*. Let us define the random variable X defined over a random space χ, with pmf $P_{\mathbf{x}}$. Given that X is a random vector, let us assume it is i.i.d.. Let us define the $\mathbf{x}_q$ quantized image obtained by plain quantization (image with no hidden message). Furthermore, let us define the QIM-stego, $\mathbf{x}_{\mathbf{QIM}}$, which is the quantized image with a hidden message. Given the cover $\mathbf{x}$ let us also define the DM-stego, $\mathbf{x}_{\mathbf{DM}}$, stego using the DM (dither modulation). The design of a parametric hypothesis test requires the probability mass function of $\mathbf{x}$, $\mathbf{x}_{\mathbf{q}}$, $\mathbf{x}_{\mathbf{QIM}}$ and $\mathbf{x}_{\mathbf{DM}}$. We are going to define all probability mass functions in terms of the pmf of $\mathbf{x}$.

Now, in the case of the plain quantization let us define the quantizer output, x_k as

$$x_k = k\Delta^*, \quad \Delta^* > 0. \tag{11}$$

Consider the following sample space. The fact that it is a sample space is to be established.

$$\Lambda_{\Delta^*} = \{x_k : x_k = k\Delta^*, k \in Z\}. \tag{12}$$

Here Z is the set of integers.

5.6. Calculating probabilities for quantized values

Let us define the random variable A with pmf, $P_{\mathbf{x}_q}$ and range equals to $\Lambda_{\Delta *}$. Let us define the quantization set

$$\chi(a, \Delta^*) \triangleq [a - \Delta^*/2, a + \Delta^*/2), \quad a \in \Lambda_{\Delta^*}. \tag{13}$$

The probability

$$P_{\mathbf{x}_q}(a) = \sum_{x \in \chi} P_{\mathbf{x}}(x)\delta_{\chi(a,\Delta^*)}(x) \tag{14}$$

with indicator function

$$\delta_{\chi(a,\Delta^*)}(x) = \begin{cases} 1 & \text{if} \quad x \in \chi(a, \Delta^*), \\ 0 & \text{if} \quad x \notin \chi(a, \Delta^*). \end{cases}$$

Let us now define a new choice of quantizers to hide binary data, $\beta = \{0, 1\}^N$. Here N is regarded as the length of a string of information (a message). We segregate the original quantizer into 2 ordinary subsets, each with step-size $\Delta = 2\Delta^*$. Let us define the set

$$\chi(s, \Delta) = [s - \Delta/2, s + \Delta/2) \tag{15}$$

Two association can be made. One quantizer associated with routing 1 is identical to that as for 0, but shifted by $\Delta/2$. Assuming the probability of 0 is equal to that of 1, we have

$$P_{\mathbf{x}_{QIM}}(s) = \frac{1}{2}\sum_{x \in \chi} P_{\mathbf{x}}(x)\delta_{\chi(s,\Delta)}(x) \tag{16}$$

where $\delta_{\chi(s,\Delta)}$ is the indicator function.

For dither modulation, we let D be a pseudo random variable uniformly distributed over $[-\Delta/4, \Delta/4)$ so that the output will cover all the values of the input, and will not leave tell-tale signs of quantization. In this range,

$$P_{\mathbf{x}_{DM}}(x) = \int_a^b \frac{2}{\Delta}dt = \frac{2(b-a)}{\Delta} = \frac{2\epsilon}{\Delta}, \quad \epsilon = b - a, \tag{17}$$

and $a, b \in [-\Delta/4, \Delta/4]$. With this dithering, any x_{DM} is valid, subject to the granularity of the system. For every received $\mathbf{x}_{DM}$ there is one and only only one valid value of d that could have made that value of x_{DM}.

For any valid x_{DM},

$$\begin{aligned}
P_{\mathbf{x_{DM}}}(x_{DM}) &= P(\{B=0,1\} \text{ and } \chi(x_{DM},\Delta) \text{ and } \{d \text{ required}\}) \\
&= P(\{B=0,1\})P_{\mathbf{x}}(\chi(x_{DM},\Delta))P_D(\{d \text{ required}\}) \\
&= \tfrac{1}{2}P_{\mathbf{x}}(\chi(x_{DM},\Delta))\tfrac{2\epsilon}{\Delta} \\
&= \tfrac{\epsilon}{\Delta}P_{\mathbf{x}}(\chi(x_{DM},\Delta))
\end{aligned}$$

5.7. Effect of embedding

Let us define n_i and n_i* as the frequency of color indices before and after embedding respectively. The relation below holds

$$|n_{2i} - n_{2i+1}| \geq |n_{2i}^* - n_{2i+1}^*|, \tag{18}$$

which means that the difference between adjacent frequency color values is reduced by the embedding process. In general and clearly we can state that for $n_{2i} > n_{2i+1}$ the bits of the hidden message change n_{2i} to n_{2i+1} more often than the other way around.

5.8. Expected Value Estimate

Instead of using color frequencies, we are going to use the DCT coefficients. The distortion will be measured by the use of the χ^2-test. Let n_i be the frequency of DCT coefficient i in the image. Because the test will uses the stego-image, the expected distribution y_i^* for the test has to be computed from the image. As a result the arithmetic mean is to be used, i.e,

$$y_i^* = \frac{n_{2i} + n_{2i+1}}{2} \tag{19}$$

to determine the empiric distribution. This average is an estimate of the expected value. The expected distribution is compared against the observed distribution $y_i = n_{2i}$.

The statistic obtained has a χ^2 distribution and is defined by

$$\chi^2 = \sum_{i=1}^{v+1} \frac{(y_i - y_i^*)^2}{y_i^*}, \tag{20}$$

where v are the degrees of freedom. Large values of χ^2 suggest a nonrandom condition or a low level of randomness. As a consequence the source is probably an original one. The opposite, small values indicate a high degree of randomness, which is often connected to encrypted hidden information.

The probability p of embedding is given by

$$p = Pr(X > \chi^2), \tag{21}$$

where $X \sim \chi^2_v$. We can compute this probability for particular regions in the image.

Now, let us partition the image in K regions with appropriate dimensions there are many approaches to take from here. We can explore the sample mean

$$\bar{X} = \frac{\sum_{i=1}^{K} \chi_i^2}{K},$$

with the assumption that $\chi_i^2 \sim \chi_{v_i}^2$ are independent random variables with mean v_i and variance $2v_i$. Note that $\bar{X} \sim N(\mu, \sigma^2)$, where

$$\mu = \frac{\sum_{i=1}^{K} v_i}{K} \quad \text{and} \quad \sigma^2 = \frac{\sum_{i=1}^{K} 2v_i}{K^2}$$

and that K is relatively large to induce normality. A special case can considered when χ_i^2 are iid.

This new consideration defines a particular framework

$$\mu = v \quad \text{and} \quad \sigma^2 = \frac{2v}{K}$$

6. Entropy

Let X a random variable with a set possible outcomes $\Lambda_X = \{a_1, \ldots, a_I\}$, with corresponding probabilities $P_X = \{p_1, \ldots, p_I\}$. The entropy of X is defined by

$$En(X) = \sum_{x \in \Lambda_X} -Pr(x) \log Pr(x),$$

with the convention for

$$Pr(x) = 0, \quad 0 \times \log 1/0 \equiv 0$$

since $\lim_{\theta \to 0^+} \theta \log 1/\theta = 0$.

Example 1

In an image with uniform distribution of gray-level intensity, let us define $\Lambda_X = \{0, \ldots, 255\}$. The corresponding probability (uniform) is given by $p_i = 1/256$ for $i = 1, \ldots, 256$. Therefore, (by substituting $\log by \log_2$), the entropy is given by

$$\begin{aligned} En(X) &= \textstyle\sum_{i=1}^{256} Pr(x_i) \log_2(1/Pr(x_i)) \\ &= 256 \times 1/256 \times \log_2(256) \\ &= 8 \log_2(2) \\ &= 8. \end{aligned}$$

Example 2

Analogously, in a RGB image, we can show that the entropy is 24.

Part IV. References

7. References

1 Hafiz Malik, K.P. Subbalakshmi and R. Chandramouli, "Nonparametric Steganalysis of QIM Data Hiding using Approximate Entropy", Proc. SPIE 6819, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X, 681914 (February 26, 2008);

2 Niels Provos and Peter Honeyman, "Detecting Steganographic Content on The Internet", CITI Technical Report 01-11, 2001

3 K. Sullivan, Z. Bi, U. Madhow, S. Chandrasekaran and B.S. Manjunath, "Steganalysis of Quantization Index Modulation Data Hiding", Image Processing, 2004. ICIP '04. 2004 International Conference on, Vol. 2, Pgs. 1165 - 1168, ONR # N0014-01-1-0380.

4 Brian Chen and Gregory W. Wornell, "Quantization Index Modulation: A Class of Provably Good Methods for Digital Watermarking and Information Embedding", IEEE Transaction on Information Theory, Vol. 47, No. 4, May 2001

Part V. Appendix

8. Appendix

Consider closed $n-$dimensional intervals $I = \{\mathbf{x} : a_j \leq x_j \leq b_j, j = 1, \ldots, n\}$ and their volumes $v(I) = \prod_{j=1}^{n}(b_j - a_j)$. To define the outer measure of an arbitrary subset E of R^n, cover E by a countable collection S of intervals I_k, and let

$$\sigma(S) = \sum_{I_k \in S} v(I_k). \tag{22}$$

Definition 3 (Lebesgue outer measure (or exterior measure)) *The Lebesgue outer measure of E, denoted $|E|_e$, is defined by*

$$|E|_e = \inf \sigma(S), \tag{23}$$

where the infimum is taken over all such covers S of E. Thus, $0 \leq |E|_e \leq +\infty$.

Definition 4 (Lebesgue Measurable or Measurable) *A subset $E \subset R^n$ is said to be $Lebesgue\ measurable$, or simply $measurable$, if given $\epsilon > 0$, there exists an open set G such that*

$$E \subset G \quad and \quad |G - E|_e < \epsilon. \tag{24}$$

Let f be a real-valued function defined on a set E in R^n, that is $-\infty \leq f(\mathbf{x}) \leq +\infty$, $\mathbf{x} \in E$. then f is called a Lebesgue measurable function on E, or simply a measurable function, if fro every finite a, the set

$$\{\mathbf{x} \in E : f(\mathbf{x}) > a\}$$

is a measurable subset of R^n. We may use the notation $\{f > a\}$ for $\{\mathbf{x} \in E : f(\mathbf{x}) > a\}$.

Definition 5 (Banach Space) *A set X is called a $Banach$ space over the complex numbers if it satisfies the following conditions:*

1. *X is a linear space over the complex numbers $\Im$; that is , if $x, y \in X$ and $\alpha \in \Im$, then $x + y \int X$ and $\alpha x \in X$.*

2. *X is a normed space; that is, for every $x \in X$ there is a non-negative number $||x||$ such that*

   a) *$||x|| = 0$ if and only if x is the zero element in X.*

   b) *$||\alpha x|| = |\alpha|||x||$ for $\alpha \in \Im$, $x \in R$.*

c) $||x+y|| \leq ||x|| + ||y||$.
If this conditions are fulfilled, $||x||$ *is called the norm of* x.

3. *X is complete with respect to its norm; that is, every Cauchy sequence in X converges in X, or if $||x_k - x_m|| \to 0$ as $k, m \to \infty$, then there is an $x \in X$ such that $||x_k - x|| \to 0$.*

If E is a measurable subset of R^n and p satisfies $0 < p < \infty$, then $L^p(E)$ denotes the collection of measurable functions f for which $\int_E |f|^p$ is finite, that is

$$L^p(E) = \left\{ f : \int_E |f|^p < +\infty \right\}, \quad 0 < p < \infty. \tag{25}$$

So from here we can mention L^1 and L^2 classes. L may be used instead of L^1.

Definition 6 (Linear Functional) *If B is a Banach space (or more generally, a normed linear space) over the real numbers, a real-valued linear functional l on B is by definition a real valued function $l(f)$, $f \in B$, which satisfies, linearity, i.e.,*

$$l(f_1 + f_2) = l(f_1) + l(f_2), \quad \text{and} \quad l(\alpha f) = \alpha l(f), \quad -\infty < \alpha < \infty.$$

Integration is an example of a linear functional:

$$I(x) = \int_a^b x(t)dt,$$

where $x(t)$ is an integrable function defined on the interval (a, b). We can defined linear functional by using different kernels, which determine transforms properties. Integral transforms are often used for the reduction of complexity of mathematical problems.

One of the most known and used linear functional transformations are the Fourier Transform.

Definition 7 (Fourier Transform) *For $f \in L(R)$, define the Fourier transform $\hat{f}$ of f by*

$$\hat{f}(x) = \int_{-\infty}^{\infty} f(t)e^{-itx}dt, \quad (x \in R). \tag{26}$$

Definition 8 (Inner product) *For $f, g \in L^2$, the inner product of f and g is defined by*

$$\langle f, g \rangle = \int f\bar{g}, \tag{27}$$

where $\bar{g}$ denotes the complex conjugate of g. The norm of f is given by $||f||^2 = \langle f, f \rangle$.

Definition 9 (Orthogonal and orthonormal) *If $\langle f, g \rangle = 0$, f and g are said to be orthogonal. A set $\{\phi_\alpha\}_{\alpha \in A}$ is orthogonal if any two of its elements are orthogonal. Furthermore, $\{\phi_\alpha\}_{\alpha \in A}$ is orthonormal if it is orthogonal and $||\phi|| = 1$.*

A particular case of the Fourier transform is the following

$$\hat{F}(x) = \hat{f}(2\pi x) = \int_{-\infty}^{\infty} f(t)e^{-i2\pi tx}dt. \tag{28}$$

where the kernel is given by, $k(x,t) = e^{-i2\pi tx}$, and $2\pi x$ is the angular frequency.

Early in the 1800s a French mathematician Joseph Fourier introduced what was called the Fourier series. Given any orthonormal system $\{\phi_k\}$ for L^2. If $f \in L^2$, the numbers defined by

$$c_k = \langle f, \phi_k \rangle \tag{29}$$

are called the Fourier coefficients of f with respect to $\{\phi_k\}$. The series $\sum_k c_k\phi_k$ is called the Fourier series of f with respect to $\{\phi_k\}$ and is denoted by $S[f]$.

A system of complex-valued functions $\{\phi_\alpha(x)\}$, all in $L^2(E)$, is called orthogonal over E if

$$\langle \phi_\alpha, \phi_\beta \rangle = \int_E \phi_\alpha \bar{\phi}_\beta \qquad \begin{cases} = 0, & \alpha \neq \beta, \\ > 0, & \alpha = \beta. \end{cases} \tag{30}$$

Note that $\langle \phi_\alpha, \phi_\beta \rangle = 1$ for all α, the orthogonal system is called normal, or orthonormal. If $\{\phi_\alpha\}$ is orthogonal, the system

$$\{\phi_\alpha / ||\phi_\alpha||_2\}$$

is orthonormal.

$$\int_E \phi_\alpha \bar{\phi}_\beta = \begin{cases} 0, & k \neq l, \\ \lambda_k > 0, & k = l. \end{cases} \tag{31}$$

Given any complex-valued $f \in L^2(E)$, we call the numbers

$$c_k = \frac{1}{\lambda_k} \int_E f\bar{\phi} \tag{32}$$

the Fourier coefficients.
We shall now consider a special orthogonal system, the trigonometric system. This name is given to the system of functions

$$e^{ikx} = \cos kx + i \sin kx, \quad k = 0, \pm 1, \pm 2, \ldots. \tag{33}$$

Note furthermore, that

$$\frac{1}{2}, \frac{e^{ikx} + e^{-ikx}}{2}, \frac{e^{ikx} - e^{ikx}}{2i}, \ldots, \quad k = 1, 2, \ldots$$

are orthonormal over any interval Q of length 2π, or what is the same thing, the functions

$$\frac{1}{2}, \cos x, \sin x, \ldots, \cos kx, \sin kx, \ldots,$$

are orthogonal over any interval Q of length 2π.

Note that for $f \in L(Q)$ can be developed into a new Fourier series

$$f \; \frac{1}{2}a_0 + \sum_{k=1}^{\infty}(a_k \cos kx + b_k \sin kx), \tag{34}$$

where

$$a_0 = (\pi/2)^{-1} \int_Q \frac{1}{2} f = \frac{1}{\pi} \int_Q f, \tag{35}$$

$$a_k = \frac{1}{\pi} \int_Q f(t) \cos ktdt, \qquad b_k = \frac{1}{\pi} \int_Q f(t) \sin ktdt \tag{36}$$

The numbers $\{a_k\}$ and $\{b_k\}$ are called the Fourier cosine and sine coefficients of f, respectively.

Note that if $Q = (-\pi, \pi)$ and f is an even function, i.e., if $f(-x) = f(x)$, the

$$a_k = \frac{2}{\pi} \int_0^{\pi} f(t) \cos ktdt, \quad b_k = 0.$$

Now, if f is an odd function, i.e., if $f(-x) = -f(x)$, then

$$a_k = 0, \quad b_k = \frac{2}{\pi} \int_0^{\pi} f(t) \sin ktdt.$$

Restricting the complex part we obtain the Fourier cosine transform (FCT), by using the real part of the complex kernel

$$Re[e^{iwt}] = \cos(\omega t) = \frac{1}{2}\left[e^{i\omega t} + e^{-i\omega t}\right]. \tag{37}$$

So the Fourier cosine transform of real or complex valued function $f(t)$, which is defined over non-negative values with non-negative angular frequency, ω, as

$$F[f](\omega) = \int_0^{\infty} f(t) \cos \omega tdt. \tag{38}$$

Now for the $discrete$ case, a particular case of the Fourier transform is given by

$$\hat{f}(n) = \sum_{k=0}^{N-1} c_k e^{i2\pi kn/N} \tag{39}$$

where c_k are Fourier coefficients defined as

$$c_k = \frac{1}{N} \sum_{k=0}^{N-1} f(n) e^{-i2\pi kn/N}. \tag{40}$$

This is called the Discrete Fourier transform (DFT).

The $multidimensional$ transforms are a simple extension case of the on dimensional. The 2-dimensional DFT is defined as

$$f(x) = \sum_{n_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} x(n_1, n_2) e^{-i2\pi\left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2}\right)}, \tag{41}$$

$$k_1 = 0, \ldots, N_1 - 1, \quad k_2 = 0, \ldots, N_2 - 1.$$

For the discrete case, Discrete Cosine Transform (DCT) can be obtained from the DFT. At this point we have to point out that f must be an even function, defined over $(-\infty, \infty)$, symetrically. There are four types of DCT. First defining the coefficients matrix we have the following types.

Type I.

$$M_I = \left(\frac{2}{N}\right)^{1/2} \left[k_m k_n \cos\left(\frac{mn\pi}{N}\right)\right], \quad m, n = 0, 1, \ldots, N.$$

Type II.

$$M_{II} = \left(\frac{2}{N}\right)^{1/2} \left[k_m \cos\left(\frac{m(n + 1/2)\pi}{N}\right)\right], \quad m, n = 0, 1, \ldots, N - 1.$$

Type III.

$$M_{III} = \left(\frac{2}{N}\right)^{1/2} \left[k_n \cos\left(\frac{(m + 1/2)n\pi}{N}\right)\right], \quad m, n = 0, 1, \ldots, N - 1.$$

Type IV.

$$M_{IV} = \left(\frac{2}{N}\right)^{1/2} \left[k_n \cos\left(\frac{(m + 1/2)(n + 1/2)\pi}{N}\right)\right], \quad m, n = 0, 1, \ldots, N - 1.$$

where

$$k_j = \begin{cases} 1 & if \quad j \neq 0 or N \\ \frac{1}{\sqrt{2}} & if \quad j = 0 or N. \end{cases}$$

Type II is the most frequently used in many applications.

The matrix obtained from the Type II is shown below.

Note that this matrix is an orthogonal matrix, i.e., its inverse equals its transpose. Name the matrix in the figure above T.

```
           [,1]        [,2]        [,3]        [,4]        [,5]        [,6]        [,7]         [,8]
[1,] 0.35355339  0.35355339  0.35355339  0.35355339  0.35355339  0.35355339  0.35355339   0.35355339
[2,] 0.49039264  0.41573481  0.27778512  0.09754516 -0.09754516 -0.27778512 -0.41573481  -0.49039264
[3,] 0.46193977  0.19134172 -0.19134172 -0.46193977 -0.46193977 -0.19134172  0.19134172   0.46193977
[4,] 0.41573481 -0.09754516 -0.49039264 -0.27778512  0.27778512  0.49039264  0.09754516  -0.41573481
[5,] 0.35355339 -0.35355339 -0.35355339  0.35355339  0.35355339 -0.35355339 -0.35355339   0.35355339
[6,] 0.27778512 -0.49039264  0.09754516  0.41573481 -0.41573481 -0.09754516  0.49039264  -0.27778512
[7,] 0.19134172 -0.46193977  0.46193977 -0.19134172 -0.19134172  0.46193977 -0.46193977   0.19134172
[8,] 0.09754516 -0.27778512  0.41573481 -0.49039264  0.49039264 -0.41573481  0.27778512  -0.09754516
```

Now in order to obtain DCT from a 8×8 matrix, M, and under the assumption that the range of M is appropriately symmetric about the origin. The DCT matrix for M is given by

$$D = TMT'$$

For example, given a picture and extracting a block 8×8, we un-normalized so that this can bring the images values to the range [0,256]. Once this is done, the resulting un-normalized must be level off by 128. The latter provides the symmetry required upon the images values.

These values are called the un-quantized DCT coefficients and our matrix is in the DCT domain.

In order to achieve the DCT matrix over the image let us define the following matrices

$$T_l = \begin{bmatrix} T & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & T & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & T & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & T \end{bmatrix}_l \quad \text{and} \quad T_r = \begin{bmatrix} T & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & T & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & T & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & T \end{bmatrix}_r ,$$

where $l = \frac{\text{\# of rows of the image}}{8}$ and $r = \frac{\text{\# number of columns}}{8}$. There are precisely l T blocks in the diagonal of T_l and r T blocks in the diagonal of T_r. Given an image I that has been un-normalized and level-off, with dimensions mod 8 = 0, the matrix

$$D = T_l I T_r', \tag{42}$$

is the DCT matrix of the image to which the DCT method has been applied for each 8×8 block.

9. R-Codes

In the previous section we compared an original image with different modifications, by having a bit switched over. We have managed to accomplish this using the R-language (2.14,2.15). We have loaded packages jpeg, ReadImage, boolfun, RGraphics and many others. The two crucial packages were ReadImages and boolfun to use read.jpeg and toBin, respectively. The first allows us to read an image in jpg or jpeg format. The second one gives an integer binary representation. Another function used from boolfun is toInt, which returns an integer from a given binary representation.

The function below, imageManipulation2, switches the i^{th} bit for each channel in every pixel of an image in jpeg or jpg format. When the picture is read, a 3-dimensional array is built. Each array contains the numeric representation for each color in every pixel. The values in this matrix are real numbers between 0 and 1. These numbers are normalized and they are of the form $n/255$, where $n = 0, 1, ..., 255$. These number are multiplied by 255 resulting in an integer ranging from 0 to 255, precisely 256 values. The integers are converted to their binary representation. A particular bit is switched over for each binary. This sequence is converted to an integer and normalized. The resulting matrix is an image different from the original.

```
%x is the image
%This function switches the bit at position n
%N is the length of the binary representation
%t is a title
%normalization (division by 255)

function (x,n,N,t)
{
    nrow <- dim(x)[1]
    ncol <- dim(x)[2]
    ncha<- dim(x)[3]
    for(i in 1:ncha)
    {
      for(j in 1:nrow)
      {
        for(k in 1:ncol)
        {
          if(n<=N && n>=1)
          {
              m = x[j,k,i]*255
              %toBin returns a binary representation of length N
              y = toBin(m,N)
              y[n] = !y[n]
              %toInt returns an integer from a binary representation of a certain length
              x[j,k,i] = toInt(y)/255
          }
        }
      }
    }
    plot(x,main=t)
    x
}

> EmbedMessageImage
function (I,m)
```

```
########################################
#EmbedMessageImage
#This function embed a message (m) of
#suitable length in an image (I)
#The embedding is achieved from the
#index {(2,1),(2,nc)} to
#{(nr-1,1),(nr-1,nc)} for each channel
########################################
{
#The message m is splitted in an array
s <- unlist(strsplit(m,split=NULL))
#The length is assigned to l
l <- length(s)
#A copy J of I is made
J <- I
#The dimensions are assigned to nr (rows), nc (cols), nh (channels)
nr <- dim(J)[1]
nc <- dim(J)[2]
nh <- dim(J)[3]
#n is assigned the dimension of the square where the message is to be written
n <- (nr-2)*nc
#if the length of the message is <= the total dimension of the image (including all 3
channels)
if(l<=n*nh)
{
#k is assigned the reminder n = l mod k
k <- l #f is assigned the minimum number of channels needed to embed the message
f <- (l-k)/n+1
#in the loop, the message is embeded by channel and message segment
for(i in 1:f)
{
#j is assigned the length of the message segment to be embedded
j <- n*(i<f)+k*(i==f)
#mx is assigned the message segment in a string of characters, not an array
mx <- Paste(s[(n*(i-1)+1):(n*(i-1)+j)])
#embedding per channel per corresponding message segment
J[2:(nr-1),,i] <- EmbedMessageMatrix(J[2:(nr-1),,i],mx)
}
}
#once the message has been embedded, the message length is embedded as well
EmbedMessageLength(J,l)
}

> EmbedMessageMatrix
function (x,m)
########################################
```

```
#EmbedMessageMatrix
#This function embed a message m, in a
#matrix with dimensions (r x c). The
#embedding is achieved sequentially,
#starting at (1,1) in the matrix
#(channel) x.
##########################################
{
#The message is converted from a string to an array of characters
s <- charToInt(unlist(strsplit(m,split=NULL)))
l <- length(s)
#An un-normalized copy of x is made
y <- x*255
nr <- dim(y)[1]
nc <- dim(y)[2]
#if the length of the message is suitable we proceed
if(l<=nr*nc)
{
#k is the reminder nc = l mod k
k <- l #f is the minimum number of rows needed to embed m
f <- (l-k)/nc+1
#The message is embeded row by row
for(i in 1:f)
{
j <- k*(i==f)+nc*(i<f)
n <- (i-1)*nc
y[i,1:j] <- s[(n+1):(n+j)]
}
}
#y is normalized
y/255.0
}

> EmbedMessageLength
function (I,n)
##########################################
#EmbedMessageLength
#This function embed the length of the
#message in the image. The embedding
#is achieved by converting the length
#into a character string, "123456".
#Then, this string is splitted as an
#array of characters, "1","2",...,"6".
#The, this is converted in integers
#according to the ascii code. The
#resulting numbers are written orderly
```

```
#on the channels’ corners, accordingly
#to the following coordinates
#(1,1),(1,nc),(nr,1),(nr,nc). Here, nr
#is the number of rows and nc is the
#number of columns.
########################################
{
J <- I*255
nr <- dim(J)[1]
nc <- dim(J)[2]
s <- unlist(strsplit(as.character(n),split=NULL))
l <- length(s)
s <- s[l:1]
for(i in 1:l)
{
if(i==1)
{
J[1,1,1] <- charToInt(s[i])
}
else if(i==2)
{
J[1,nc,1] <- charToInt(s[i])
}
else if(i==3)
{
J[nr,1,1] <- charToInt(s[i])
}
else if(i==4)
{
J[nr,nc,1] <- charToInt(s[i])
}
else if(i==5)
{
J[1,1,2] <- charToInt(s[i])
}
else if(i==6)
{ J[1,nc,2] <- charToInt(s[i])
}
else if(i==7)
{ J[nr,1,2] <- charToInt(s[i])
}
else if(i==8)
{ J[nr,nc,2] <- charToInt(s[i])
}
else if(i==9)
{ J[1,1,3] <- charToInt(s[i])
```

```
}
else if(i==10)
{ J[1,nc,3] <- charToInt(s[i])
}
else if(i==11)
{ J[nr,1,3] <- charToInt(s[i])
}
else if(i==12)
{ J[nr,nc,3] <- charToInt(s[i])
}
}
if(l<12)
{
for(k in (l+1):12)
{ if(k==1)
{ J[1,1,1] <- charToInt("0")
}
else if(k==2)
{ J[1,nc,1] <- charToInt("0")
}
else if(k==3)
{ J[nr,1,1] <- charToInt("0")
}
else if(k==4)
{ J[nr,nc,1] <- charToInt("0")
}
else if(k==5)
{ J[1,1,2] <- charToInt("0")
}
else if(k==6)
{ J[1,nc,2] <- charToInt("0")
}
else if(k==7)
{ J[nr,1,2] <- charToInt("0")
}
else if(k==8)
{ J[nr,nc,2] <- charToInt("0")
}
else if(k==9)
{ J[1,1,3] <- charToInt("0")
}
else if(k==10)
{ J[1,nc,3] <- charToInt("0")
}
else if(k==11)
{ J[nr,1,3] <- charToInt("0")
}
else if(k==12)
{ J[nr,nc,3] <- charToInt("0")
}
}
}
J/255.0
}

> ExtractMessageImage
function (I)
#########################################
#ExtractMessageImage
#This function extract a message from
#an image. It is assumed that the
#message is embedded accordingly to the
#algorithm employed in
#EmbedMessageImage.
#########################################
{
nr <- dim(I)[1]
nc <- dim(I)[2]
nh <- dim(I)[3]
l <- ExtractMessageLength(I)
n <- (nr-2)*nc
k <- l f <- (l-k)/n+1
s <- ""
for(i in 1:f)
{
j <- n*(i<f)+k*(i==f)
s <- paste0(s,ExtractMessageMatrix(I[2:(nr-1),,i],j))
}
s
}

> ExtractMessageMatrix
function (x,l)
#########################################
#ExtractMessageMatrix
#This function extract a message from
#a matrix. It is assumed that the
#message is embedded accordingly to the
#algorithm employed EmbedMessageMatrix.
#########################################
{
Paste(intToChar(c(t(x))[1:l]*255))
```

```
}

> ExtractMessageLength
function(I)
########################################
#ExtractMessageLength
#This function extract the length from
#an image. It is assumed that the
#length of the message is embedded
#accordingly to the algorithm employed
#in EmbedMessageLength.
########################################
{
J <- I*255
nr <- dim(J)[1]
nc <- dim(J)[2]
s <- intToChar(J[1,1,1])
s <- paste(intToChar(J[1,nc,1]),s,sep="")
s <- paste(intToChar(J[nr,1,1]),s,sep="")
s <- paste(intToChar(J[nr,nc,1]),s,sep="")
s <- paste(intToChar(J[1,1,2]),s,sep="")
s <- paste(intToChar(J[1,nc,2]),s,sep="")
s <- paste(intToChar(J[nr,1,2]),s,sep="")
s <- paste(intToChar(J[nr,nc,2]),s,sep="")
s <- paste(intToChar(J[1,1,3]),s,sep="")
s <- paste(intToChar(J[1,nc,3]),s,sep="")
s <- paste(intToChar(J[nr,1,3]),s,sep="")
s <- paste(intToChar(J[nr,nc,3]),s,sep="")
unlist(as.numeric(s))
}

> EmbedMessageLSB
function (I,m)
############################################
#EmbedMessageLSB
#This function embed a message in an image
#using the LSB technique.
############################################
{
s <- binaryRep(m)
l <- length(s)
m <- dim(I)[1]
n <- dim(I)[2]
if(l <= 3*(m-2)*n)
{
J <- EmbedMessageLength(I,l)
k <- ceiling(l/3)
J[2:(m-1),,1] <- EmbedMessageMatrixLSB(J[2:(m-1),,1],s[1:k])
if(k < l)
{
J[2:(m-1),,2] <- EmbedMessageMatrixLSB(J[2:(m-1),,2],s[(k+1):(2*k)])
if(2*k < l)
{
J[2:(m-1),,3] <- EmbedMessageMatrixLSB(J[2:(m-1),,3],s[(2*k+1):l])
}
}
}
else
{
J <- "Error: length of message surpass image dimension
}
J
}

> EmbedMessageMatrixLSB
function (i,s)
##########################################
#EmbedMessageMatrixLSB
#This function embed a message into a
#channel.
##########################################
{
l <- length(s)
m <- nrow(i)
n <- ncol(i)
k <- floor(m*n/l)
ind <- seq(1,m*n,k)[1:l]
jj <- c(t(i))
j <- ToBin(jj[ind]*255,8)
j[,8] <- s
jj[ind] <- ToInt(j)/255.0
JJ <- matrix(jj,ncol=n,nrow=m,byrow=TRUE)
JJ
}

function (I)
############################################
#ExtractMessageLSB
#This function extract a message from a
#channel using the LSB technique.
############################################
{
```

```
l <- ExtractMessageLength(I)
m <- dim(I)[1]
n <- dim(I)[2]
k <- ceiling(l/3)
k1 <- floor((m-2)*n/k)
ind1 <- seq(1,(m-2)*n,k1)[1:k]
M <- ToBin(c(t(I[2:(m-1),,1]))[ind1]*255,8)[,8]
if(k < l)
{
k2 <- k1
ind2 <- ind1
M <- c(M,ToBin(c(t(I[2:(m-1),,2]))[ind2]*255,8)[,8])
if(2*k < l)
{
k3 <- floor((m-2)*n/(l-2*k))
ind3 <- seq(1,(m-2)*n,k3)[1:(l-2*k)]
M <- c(M,ToBin(c(t(I[2:(m-1),,3]))[ind3]*255,8)[,8])
}
}
Paste(intToChar(ToInt(matrix(M,nrow = l/8,ncol=8,byrow=TRUE))))
}

function (m)
##########################################
#binaryRep
#This function returns a binary string
#representing the message m. First, each
#character is converted to an integer
#(ascii code). These integers are
#converted to binary strings of length 8
#returning a string of all binary rep
#in an array.
##########################################
{
c(t(ToBin(charToInt(unlist(strsplit(m,split=NULL))),8)))
}

> ToInt
function (M)
##########################################
#ToInt
#ToInt is a modified version of toInt. This accepts arrays of binary strings
##########################################
{
res <- c()
for (i in 1:nrow(M))
```

```
{
res <- c(res,toInt(M[i,]))
}
res
}

> ToBin
function (a, n)
########################################
#ToBin
#ToBin is a modified version of toBin. This accepts arrays of binary strings
########################################
{
a <- as.integer(a)
n <- as.integer(n)
if (n == 0)
res <- c()
else {
res <- matrix(nrow=length(a), ncol=n)
for (i in 1:n)
{
res[,i] <- aa <- a}
}
res
}
```

```
bc2p
function (y,d,mn,xl,yl,ty)
{
	nr <- dim(x)[1]/8
	nc <- dim(x)[2]/8
	V <- array(dim=c(nr,nc,3))
	x <- DCTc(y)
	for(i in 1 nr)
	{
		for(j in 1:nc)
		{
			ri <- 8*(i-1)+1
			rf <- 8*i
			ci <- 8*(j-1)+1
			cf <- 8*j
			dctc8 <- c(x[ri:rf,ci:cf])
			m <- min(dctc8)
			M <- max(dctc8)
			k1 <- floor((m+d/2)/d)
			k2 <- ceiling((M-d/2)/d)
			cc <- c()
			for(l in k1:k2)
			{
				cc <- c(cc,length(chi(dctc8,l,d)))
			}
			v <- Chi2Prob(cc)
			for(k in 1:3)
			{
				V[i,j,k] <- v[k]
			}
		}
	}
	plot(V[,,3][V[,,3]!="NaN"],type=ty,main=mn,xlab=xl,ylab=yl)
}

bc2ptest
function (y,d)
{
	nr <- dim(y)[1]/8
	nc <- dim(y)[2]/8
	V <- array(dim=c(nr,nc,3))
	x <- DCTc(y)
	for(i in 1 nr)
	{
		for(j in 1:nc)
		{
			ri <- 8*(i-1)+1
			rf <- 8*i
			ci <- 8*(j-1)+1
			cf <- 8*j
			if(d!=0)
			{
				k1 <- freqlowerbound(min(x[ri:rf,ci:cf]),d)
				k2 <- frequpperbound(max(x[ri:rf,ci:cf]),d)
```

```
                }
                cc <- c()
                for(l in k1:k2)
                {
                        cc <- c(cc,length(chi(x[ri:rf,ci:cf],l,d)))
                }
                V[i,j,1:3] <- Chi2Prob(cc)
        }
    }
    V
}

bc2ptest2
function (y,d)
{
    nr <- dim(y)[1]/8
    nc <- dim(y)[2]/8
    V <- array(dim=c(nr,nc,3))
    x <- DCTc(y)
    cc <- c()
    for(i in 1 nr)
    {
        for(j in 1:nc)
        {
                ri <- 8*(i-1)+1
                rf <- 8*i
                ci <- 8*(j-1)+1
                cf <- 8*j
                if(d!=0)
                {
                        k1 <- freqlowerbound(x[ri:rf,ci:cf],d)[2]
                        k2 <- frequpperbound(x[ri rf,ci:cf],d)[2]
                }
                for(l in k1:k2)
                {
                        cc <- c(cc,length(chi(x[ri:rf,ci:cf],l,d)))
                }
        }
    }
    cc
}

binaryRep
function (m)
########################################
#binaryRep
#This function returns a binary string
#representing the message m. First, each
#character is converted to an integer
#(ascii code). These integers are
#converted to binary strings of length 8
#returning a string of all binary rep
#in an array.
########################################
```

```
{
	c(t(ToBin(charToInt(unlist(strsplit(m,split=NULL))),8)))
}

BlockChi2Prob
function (x)
################################################
#BlockChi2Prob
#BlockChi2Prob returns the probability of
#embedding for each block
################################################
{
	nr <- dim(x)[1]/8
	nc <- dim(x)[2]/8
	V <- array(dim=c(nr,nc,3))
	dt <- DCTc(x)
	for(i in 1 nr)
	{
		for(j in 1:nc)
		{
			ri <- 8*(i-1)+1
			rf <- 8*i
			ci <- 8*(j-1)+1
			cf <- 8*j
			v <- Chi2Prob(table(dt[ri:rf,ci:cf]))
			for(k in 1:3)
			{
				V[i,j,k] <- v[k]
			}
		}
	}
	V
}

BlockChi2Prob2
function (x)
{
	nr <- dim(x)[1]/8
	nc <- dim(x)[2]/8
	V <- array(dim=c(nr,nc,3))
	for(i in 1 nr)
	{
		for(j in 1:nc)
		{
			ri <- 8*(i-1)+1
			rf <- 8*i
			ci <- 8*(j-1)+1
			cf <- 8*j
			v <- Chi2Prob2(x[ri:rf,ci:cf]*255)
			for(k in 1:3)
			{
				V[i,j,k] <- v[k]
			}
		}
```

```
    }
    V
}

BlockChi2Value
function (x)
{
    nr <- dim(x)[1]/8
    nc <- dim(x)[2]/8
    chisqvalues <- matrix(nrow=nr,ncol=nc)
    for(i in 1 nr)
    {
        for(j in 1:nc)
        {
            ri <- 8*(i-1)+1
            rf <- 8*i
            ci <- 8*(j-1)+1
            cf <- 8*j
            chisqvalues[i,j] <- Chi2Value(x[ri:rf,ci:cf]*255)
        }
    }
    chisqvalues
}

BlockChi2Value2
function (x)
{
    nr <- dim(x)[1]/8
    nc <- dim(x)[2]/8
    chisqvalues <- matrix(nrow=nr,ncol=nc)
    for(i in 1 nr)
    {
        for(j in 1:nc)
        {
            ri <- 8*(i-1)+1
            rf <- 8*i
            ci <- 8*(j-1)+1
            cf <- 8*j
            chisqvalues[i,j] <- Chi2Value2(x[ri:rf,ci:cf]*255)
        }
    }
    chisqvalues
}
```

Steganography Detection Using Entropy Measures

Eduardo Meléndez
Advisor: Dr. Jeffrey Duffany
Polytechnic University of Puerto Rico

Abstract

In this ongoing research we study the problem of detecting the existence of a hidden message in an image. JPEG is the image format to work with, however, the study shall not be limited to this type. Many techniques for detecting embedded messages have been proposed. First, we study different mechanisms of embedding and how these affected randomness. We implemented different programs developed in the R-language. Methods of embedding were considered from very simple to more complex using the LSB technique, spreading and an equal distribution of a message between channels. We study two statistical approaches for the steganalysis. First, we study the χ^2 statistical test which works on the concept adjacent DCT-coefficients frequencies. Second, we also study the likelihood ratio test (LRT) for the implementation of the Quantization Index Modulation (QIM) and the Dither Modulation (DM) to study the Least Significant Bits (LSB) embedding technique. The LSB is the focus of our study since their modification cannot be perceived by the human eye, i.e. its modification raises no significant attention. We calculate DCT-coefficients matrices using block diagonal matrices to overcome the use of loops. Some codes were written using R-language to study and obtain some results to analyze DCT coefficients frequencies. Even more, we use the concept of the QIM for the quantized values obtained by grouping the DCT coefficients.

Keywords: Steganography, steganalysis, LSB, DCT, QIM, Shannon's entropy, R language, χ^2-test

Introduction

The object of steganography is to hide the fact that some sort of communication is taking place. This is the ground for the use of the Least Significant Bit. The modification of a bit in a pixel will have an impact on the level of visual perception. As we move along the binary representation from the least significant to the more significant visually we will be able to perceive that a change has taken place. For example, Figure 2 is the result of changing the 8th bit in the binary representation for each color from Figure 1. Note that each pixel is represented by a string of 8 bits in a monochrome picture and 24 in a RGB picture. From black to white, we have $2^8 = 256$ different tones of gray. In a RGB image we have $256^3 = 16{,}777{,}216$ (256 for each color).

Now, any visual insight of the existence of a modification leads steganography failure. In reality, no such modification shall be perceptible by the human eye.

Figure 1. Original

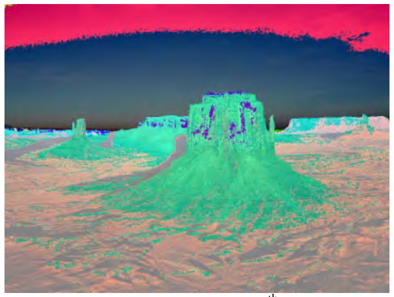


Figure 2. Modification of the 8th bit per pixel

Methodology

In order to achieve images modification we construct our own codes. Below a code written in R-language uses the LSB modification technique.

```
function (I,m){
 s <- binaryRep(m)
 l <- length(s)
 m <- dim(I)[1]
 n <- dim(I)[2]
 ch <- dim(I)[3]
 z <- ch*(m-2)*n
 if(l <= z) {
  J <- EmbedMessageLength(I,l)
  k <- ceiling(l/ch)
  J[2 (m-1),,1] <- EmbedMessageMatrixLSB(J[2:(m-1),,1],s[1:k])
  if(k < l){
    J[2:(m-1),,2]
      <- EmbedMessageMatrixLSB(J[2:(m-1),,2],s[(k+1):(2*k)])
    if((2*k) < l) {
     J[2:(m-1),,3]
      <- EmbedMessageMatrixLSB(J[2:(m-1),,3],s[(2*k+1) l])
    }
  }
 }
 else{
  J <- "Error: length of message surpass image dimension"
 }
return J
}
```

There are many approaches being used to detect stego images. Discrete Cosine Transforms play a crucial role in many of these techniques. The DCT matrix is a special case of the Discrete Fourier Transforms. A version of the DCT used in our research is

$$T_{ij} = \frac{1}{\sqrt{2N}} C(i)C(j)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} m_{xy} d_N(x,i) d_N(y,j)$$

where

$$C(n) = \begin{cases} \frac{1}{\sqrt{2N}} & if \quad n = 0 \\ 1 & if \quad n > 0 \end{cases}, \quad d_N(x,i) = cos\left[\frac{(2a+1)b\pi}{2N}\right]$$

and m_{xy} are the entries from a matrix M, and $T = (T_{ij})_{ij}$, an orthogonal matrix. We construct the DCT coefficients matrix for D_{xy} by taking the matrix product $TM_{xy}T'$. The matrix M_{xy} is being transformed previously to the DCT domain, (symmetric about 0). We also construct block diagonal matrices to obtain the image DCT coefficients. These block diagonal matrices are of the form (d = dimension squared)

$$T_d = \begin{bmatrix} T & 0 & 0 & & 0 \\ 0 & T & 0 & \cdots & 0 \\ 0 & 0 & T & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & T \end{bmatrix}_d$$

so that a particular form of DCT coefficient image is obtained by $D = T_l M T_r'$, where $D = (D_{xy})_{xy}$ and $M = (M_{xy})_{xy}$.

One approach is that the frequency of the DCT coefficients is being affected by the presence of an encrypted hidden message. Given c_i and c_i^* the frequencies of the color indices previous and after the embedding, respectively can be described by

$$|c_{2i} - c_{2i+1}| \geq |c_{2i}^* - c_{2i+1}^*|.$$

By considering the DCT coefficients frequencies

$$x = \sum_i^{v+1} \frac{(y_i - y_i^*)^2}{y_i^*}, \quad y_i^* = \frac{n_{2i} + n_{2i+1}}{2} \text{ and } y_i = n_{2i},$$

n_i the frequencies of the DCT coefficients in a region.

Given that x approximate to a χ^2 rv, we may obtain the probability of embedding $p = P(\chi^2 > x)$. Figure 3 shows the probabilities of embedding for different regions (8x8 sequential blocks) using the image in Figure 1 to embed 6Napolen.txt from http://www.textfiles.com/stories/.

To detect stego images we may use the (QIM). Given a host signal x, with pmf P_X, message m and a mapping $s(x,m)$, an ensemble of discontinuous functions have the strength of embedding by minimizing the distortion, $D(s,x) = \frac{1}{N}\|s - x\|^2$.

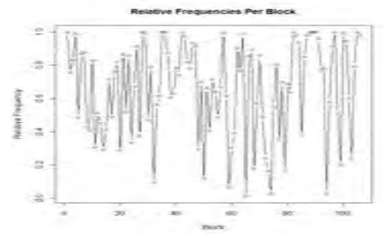


Figure 3. Probabilities per Sequential 8 x 8 Blocks

These discontinuities allow us to define the quantized values for a given $\Delta^* > 0$. $x_k = k\Delta^*$, where k is an integer. Let us define set $A_{\Delta^*} = \{k\Delta^* : k \text{ is an integer}\}$ and $\chi(a,\Delta^*) = \left[a - \frac{\Delta^*}{2}, a + \frac{\Delta^*}{2}\right)$. The pmf of the quantized cover, $x_q \in A_{\Delta^*}$, without embedding is given by $P_{X_q}(a) = \sum_{x\in\chi(a,\Delta^*)} P_X(x)$. Analogously, the pmf of the quantized embedding, $x_{QIM} \in A_\Delta$, $P_{X_{QIM}}(a) = \frac{1}{2}\sum_{x\in\chi(a,\Delta)} P_X(x)$, where $\Delta = 2\Delta^*$, $a \in A_\Delta$.

Conclusions

These probabilities lead us to the Ratio Likelihood Test and to the Shannon Entropy. Given p_b (q_b =1- p_b), the probability of embedding per block, $b \in B$ (the space of blocks), and assuming a uniform distribution, i.e., $P(b) = p$ for $b \in B$, the entropy may be obtain by

$$H(I) = -\sum_{b\in B}[p_b \, p\log\{p_b p\}] - \sum_{b\in B}[q_b \, p\log\{q_b p\}].$$

The latter result is to be studied.

Acknowledgement

* This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number: W911NF1110174.
* Also I want to thanks Dr. Alfredo Cruz, PI, for his support and opportunity to be part of this research

Reference

[1] Hafiz Malik, K.P. Subbalakshmi and R. Chandramouli, "Nonparametric Steganalysis of QIM Data Hiding using Approximate Entropy", *Proc. SPIE* 6819, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X, 681914 (February 26, 2008);
[2] Niels Provos and Peter Honeyman, "Detecting Steganographic Content on The Internet", CITI Technical Report 01-11, 2001
[3] K. Sullivan, Z. Bi, U. Madhow, S. Chandrasekaran and B.S. Manjunath, "Steganalysis of Quantization Index Modulation Data Hiding", Image Processing, 2004. ICIP '04. 2004 International Conference on, Vol. 2, Pgs. 1165 - 1168, ONR # N0014-01-1-0380.
[4] Brian Chen and Gregory W. Wornell, "Quantization Index Modulation: A Class of Provably Good Methods for Digital Watermarking and Information Embedding", IEEE Transaction on Information Theory, Vol. 47, No. 4, May 2001